A Survey of Techniques for Information Flow Analysis and Modeling of Faceted Values for Controlling Malicious JavaScript

Konstantinos Xirogiannopoulos and Benjamin Bengfort

Department of Computer Science
University of Maryland
{kostasx,bengfort}@cs.umd.edu

November 6, 2015

Proposal

Recently the integration of web services via embedded publisher-specific JavaScript has become increasingly popular to create rich web applications in a plug-and-play fashion. The typical workflow involves the inclusion of snippets of third party code into the client application, customized with APIKeys or other site-specific identifiers. When executed, the JavaScript will communicate with the web-service, often downloading additional JavaScript or CSS that is executed in the client browser.

For example, the Disqus [3] service incorporates a fully functional comment section into any web page or blog post. It does this through the inclusion of a simple 14 line JavaScript code, customized with 4 site specific configuration variables. Although security measures such as sandboxing and cross-domain exclusion attempt to protect users from cross site scripting (XSS) and injection attacks, once third party code is embedded, it is from then onward treated as such; a native part of the codebase. Moreover, though there are usually various levels of confidentiality in any publicly available code, the injection of "outsider code" makes way for potential malicious behavior.

Security flaws in the embedding of third-party libraries have been shown particularly for accessing and downloading streaming music [4]. Here sites such as Aimini and Groove Shark were exploited to retrieve music through an inspection and analysis of the injected JavaScript. This inspection revealed secure variables and endpoints that were accidentally exposed in the program itself. If there was a way to control the flow of information between variables within the code, these security flaws would not have exposed sensitive information.

Information Flow Analysis [2] of programs is a systematic approach to dealing with how information in the form of the data stored in a program's variables, gets propagated within the code itself. Information Flow Analysis methods attempt to control the way data is passed on inside the program by means of specifying a security level of each variable towards preventing the propagation of sensitive information to unsafe variables. This idea has been extended to a type system that can guarantee safety via type analysis [5].

In this survey paper, we will attempt to enumerate, study and discuss the many techniques for Information Flow Analysis of programs, as proposed by the academic community. We will compare

and contrast these techniques and offer a complete overview of the state of the art. Furthermore, inspired by the approach taken by Austin and Flanagan in [1], we will attempt to model *faceted evaluation* semantics in the *OCaml* language.

Milestones and Schedule

Below is a rough schedule we will attempt to follow for this work. We list the milestones we intend to complete as well as their tentative deadlines.

1. Study and write up an overview of the fundamentals of Information Flow Analysis by *11/14*
2. Study [1] extensively and compile a reading list by *11/21*
3. Study and enumerate Information Flow Analysis techniques by *11/28*
4. Implement a model for Faceted Evaluation in OCaml by *12/5*

References

[1] Thomas H Austin and Cormac Flanagan. Multiple facets for dynamic information flow. *ACM SIGPLAN Notices*, 47(1):165–178, 2012.

[2] Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[3] Daniel Ha and Jason Yan. Disqus.

[4] Franz Payer, Zachary Cutlip, and Craig Heffner. Exploiting Music Streaming with JavaScript. Las Vegas, NV, August 2013.

[5] Geoffrey Smith, Cynthia E. Irvine, Dennis Volpano, and others. A sound type system for secure flow analysis. 1996.